JTAG - światowy standard testowania i programowania układów cyfrowych, część 1

Przedstawiamy pierwszy w krajowej prasie technicznej, tak szczegółowy opis standardu JTAG. Standard ten nabiera coraz większego znaczenia praktycznego, gdyż większość aktualnie produkowanych cyfrowych układów scalonych zawiera w swoim wnętrzu elementy architektury JTAG. Dotyczy to zarówno struktur PLD, jak i FPGA, mikroprocesorów, mikrokontrolerów, a także układów serii 74. W ciągu kilku najbliższych lat JTAG stanie się zapewne powszechnie uznawanym standardem, ponieważ jego elastyczność pozwala na wykorzystanie go do programowania układów ISP (zwłaszcza struktur PLD i mikrokontrolerów).

Na początku lat 90. organizacja IEEE (ang. Institute of Electrical and Electronic Engineers - Instytut Inżynierów Elektryków i Elektroników) przedstawiła nową normę standaryzującą sposób testowania układów scalonych. Nosi ona nazwę IEEE 1149.1 „The Test Access Port and Boundary Scan Architecture", co można przetłumaczyć jako „Port dostępu dla testów i architektura testowania ścieżką krawędziową". Norma ta dotyczy metody umożliwiającej wprowadzanie i odczyt danych testowych do dowolnego układu cyfrowego (wykonanego zgodnie z normą JTAG) za pośrednictwem specjalnej ścieżki testowej. Pojęcie „ścieżki testowej" odpowiada pewnemu fizycznemu i logicznemu fragmentowi wnętrza układu, wydzielonemu specjalnie do celów testowania i/lub programowania układu.

Do czego jest potrzebny JTAG?

Złożoność współcześnie konstruowanych urządzeń cyfrowych rośnie w ogromnym tempie. Miliony tranzystorów integrowanych w strukturach układów scalonych tworzą bardzo rozbudowane struktury logiczne, których sprawdzenie standardowymi metodami testowymi (analiza sygnatur, kontrola reakcji urządzenia na wymuszone pobudzenia logiczne) wymaga ogromnej wiedzy od inżynierów, dużego doświadczenia, zabiera bardzo dużo czasu i nie daje zbyt dużej pewności co do otrzymanych wyników. Co więcej, współczesne procesory i układy PLD dużej skali integracji trudno jest dokładnie przetestować, ze względu na ograniczoną możliwość wyprowadzenia na zewnątrz struktury dużej liczby punktów logicznych. Do dokładnego sprawdzenia takich układów niezbędne są specjalne przyrządy pomiarowe, których ceny (ze względu na specyfikę działania) osiągają poziom setek tysięcy, a nawet milionów USD.

Alternatywą testowania „ręcznego" jest JTAG - zamiast analizy setek lub tysięcy punktów pomiarowych testowanego układu wystarczy wpisać poprzez złącze szeregowe JTAG (układy są łączone w łańcuch) odpowiedni program testowy. Wyniki działania tego programu analizuje komputer wyposażony w odpowiednie (lecz tanie, ze względu na uniwersalność) oprogramowanie. Wyniki testu można otrzymać po kilkunastu sekundach lub co najwyżej kilku minutach działania programu! W przypadku wykrycia błędu wskazywany jest nie tylko uszkodzony układ, lecz także jego wyprowadzenie, co niezwykle upraszcza usuwanie usterek.

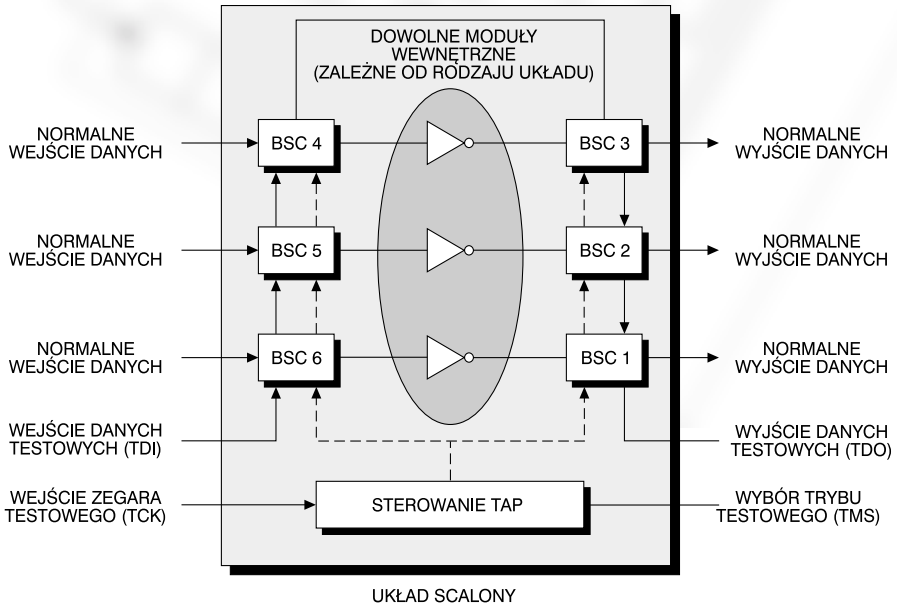
Jest to chyba wystarczający powód, aby uznać JTAG za zjawisko przełomowe w testowaniu układów programowalnych.

Powstanie standardu

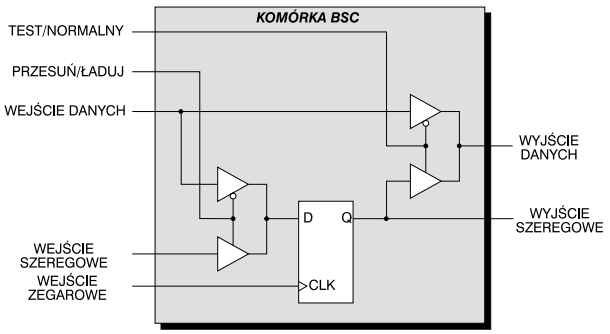
Idea standardu JTAG powstała w 1985 roku, kiedy to Frans Beenker, pracownik Philips Research Laboratories opublikował artykuł, w którym wyraził potrzebę opracowania lepszego, szybszego, w większym stopniu opartego na strukturalnym podejściu sposobu testowania złożonych układów cyfrowych. Wyraził zdecydowane przekonanie, że to właśnie na technikę testowania ścieżką krawędziową padnie wybór, gdyż umożliwia rozwiązanie wielu współczesnych i przyszłych problemów związanych z przeprowadzaniem testów.

Wkrótce po pojawieniu się artykułu Beenkera, grupa europejskich producentów poparła szybkie utworzenie standardu opisującego sposób naprawy i testowania układów cyfrowych. W wyniku tych działań powstała organizacja *Joint European Test Action Group* (JETAG - w wolnym przekładzie „Połączona europejska grupa na rzecz opracowania testu"). Wkrótce dołączyły do niej firmy amerykańskie, tworząc *Joint Test Action Group* (JTAG).

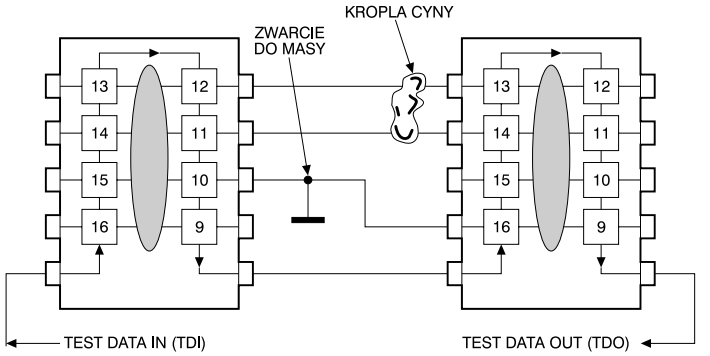
Pierwsza wersja standardu JTAG zaproponowana została w roku 1986 przez Beenkera, Chantal Vivier (Bull Systems) i Colina Maundera (British Telcom Research Labs). Później pojawiły się następne propozycje. W roku 1988 wersja 2.0 standardu JTAG została przedłożona IEEE jako propozycja międzynarodowego standardu. Organizacja



Rys. 1.



Rys. 2.



UWAGA: SEKWENCJE ROZPOCZYNAJĄ SIĘ OD LSB

BSC#	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
SEKWENCJA WEJŚCIOWA	X	X	X	X	1	0	1	X	X	X	X	X	X	X	X	X
SEKWENCJA OCZEKIWANA	X	X	X	X	X	X	X	X	1	X	0	1	X	X	X	X
SEKWENCJA OTRZYMANA	X	X	X	X	X	X	X	X	0	X	1	1	X	X	X	X
SEKWENCJA WEJŚCIOWA	X	X	X	X	0	1	0	X	X	X	X	X	X	X	X	X
SEKWENCJA OCZEKIWANA	X	X	X	X	X	X	X	X	0	X	1	0	X	X	X	X
SEKWENCJA OTRZYMANA	X	X	X	X	X	X	X	X	X	X	1	1	X	X	X	X

Rys. 4.

IEEE przyjęła go w dniu 15 lutego 1990, nadając mu numer 1149.1 i nazwę „*The Test Access Port and Boundary Scan Architecture*".

Jak są testowane układy?

Testowanie przy pomocy ścieżki krawędziowej nie wymaga fizycznego dostępu do każdego wyprowadzenia układu scalonego, aby przeprowadzić test lub zlokalizować uszkodzenie. Układ scalony, zgodny z tym standardem, posiada w swoim wnętrzu zestaw bramek logicznych tworzących specjalny łańcuch. Bramki te ulokowane są między wyprowadzeniami układu a jego wewnętrznymi układami logicznymi - stąd nazwa „testowanie krawędziowe".

Podstawowa architektura ścieżki krawędziowej przedstawiona została na **rys.1**. Prostokąty ulokowane między wyprowadzeniami układu i logiką wewnętrzną noszą nazwę komórek ścieżki krawędziowej (ang. Boundary Scan Cells - *BSC*). Komórki te są połączone w taki sposób, by powstała ścieżka między wejściem (*TDI*) i wyjściem danych testowych układu (*TDO*).

Podczas normalnej pracy sygnały wejściowe i wyjściowe są przekazywane od standardowych wejść do standardowych wyjść układu. W trybie testowania krawędziowego komórki *BSC* są sterowane w taki sposób, że z wejścia *TDI* można wprowadzić do układów wewnętrznych dane testowe przez dowolną z komórek *BSC*, znajdującą się od strony wejścia. Wejścia *TCK* i *TMS* umożliwiają równoległe sterowanie komórkami *BSC*. Sygnały wyjściowe logiki wewnętrznej zostają następnie wyprowadzone przez odpowiednie komórki *BSC* na wyjście *TDO*. Taka metoda jest przydatna do testowania wewnętrznych układów logicznych elementu scalonego.

Zewnętrzne testowanie połączeń układ-ścieżka, znajdowanie niesprawnych połączeń lutowanych lub uszkodzeń sąsiadujących układów scalonych jest dokonywane przez wyprowadzenie sygnału testowego z wyjściowych komórek *BSC* jednego układu i analizę sygnału pojawiającego się na wejściu komórek *BSC* współpracującego układu scalonego. Taki sposób testowania pozwala uniknąć wielu problemów związanych z fizycznym dostępem do wyprowadzeń układów.

BSC#	6	5	4	3	2	1
SEKWENCJA WE	1	0	1	X	X	X
SEKWENCJA WY	X	X	X	0	1	0

Uwaga: Sekwencje rozpoczynają się od LSB

Rys. 3.

Komórka *BSC*

Komórka *BSC* stanowi podstawowy element umożliwiający testowanie krawędziowe. Schemat takiej komórki przedstawia **rys.2**. Jak z niego wynika, zawiera ona przerzutnik D typu zatrzask oraz bufory trójstanowe. Bufory sterowane są sygnałami przez port dostępu testowego (*TAP*), którego działanie zostanie bardziej szczegółowo omówione dalej.

Oczywiście, struktura typowej komórki *BSC* jest zazwyczaj daleko bardziej złożona niż wynikałoby to z rys. 2. Jest tak dlatego, że wyprowadzenia układów scalonych mogą być dwukierunkowe, trójstanowe itd. Ilustracja ta ma jedynie ułatwić Czytelnikowi zrozumienie idei i sposobu wykorzystania komórki *BSC*.

Komórki *BSC* - normalny tryb pracy

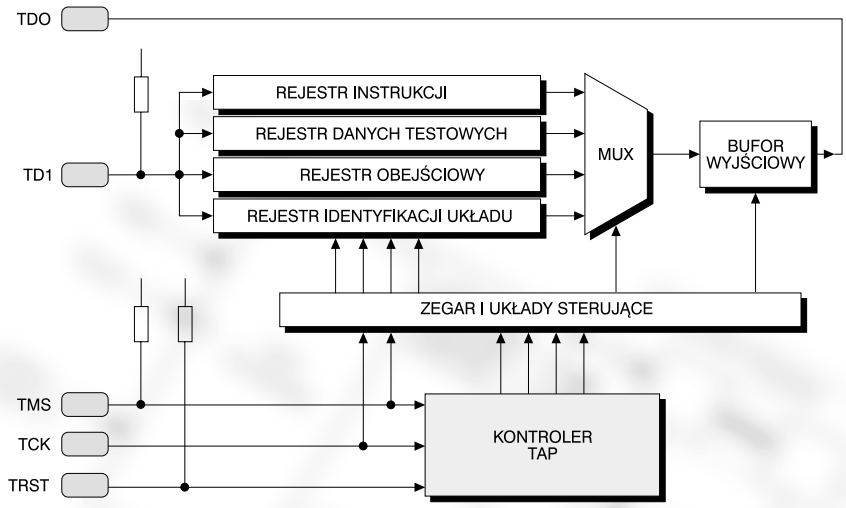
Dla celów niniejszej dyskusji załóżmy, że rys. 2 przedstawia schemat dowolnej komórki *BSC* z rys.1. Podczas normalnej pracy układu dane pochodzące z wewnętrznych układów są podawane na linię DATA INPUT komórki. Stan linii sterującej TEST/NORMAL jest niski, natomiast linii sterującej SHIFT/LOAD - wysoki. Takie warunki umożliwiają niezakłóconą transmisję danych do wyjścia DATA OUT. Cały układ scalony funkcjonuje tak, jakby komórki *BSC* nie istniały.

Podczas gdy układ scalony realizuje swe zwykłe funkcje, do komórki *BSC* można wprowadzić dane testowe lub je z niej wyprowadzić. *Wejście zegarowe* jest wykorzystywane do wprowadzenia do przerzutnika *D* danych obecnych na linii *Wejście szeregowe*. W przedstawianym przykładzie linię *Wejście szeregowe* na rys.1 stanowi linia *Wejście danych testowych*. Ponieważ linie sterowania komórek *BSC* są połączone równolegle, kolejne impulsy zegarowe będą powodować przesuwanie danych z komórki *BSC* do komórki o niższym numerze w łańcuchu komórek *BSC*. Istnieje także możliwość zapamiętania danych występujących na wejściach komórek *BSC*. Jeśli do układu *BSC* zostaną wprowadzone takie dane lub dane testowe zostaną z niego wyprowadzone lub doń wprowadzone, mówi się, że układ *BSC* jest w trybie *Sample* (próbkowania) lub *Preload* (ładowania danych).

Komórki *BSC* - tryb pracy podczas testów

Omawiając tryb testowy trzeba zdawać sobie przede wszystkim sprawę z tego, że linie sterujące wszystkich komórek są połączone równolegle. Innymi słowy, podanie impulsu zegarowego na jedną z komórek *BSC* jest równoważne podaniu go na wszystkie komórki, które mogą być albo w trybie testowym, albo w trybie zwykłym. Pamiętając o tym, można wyobrazić sobie następujące działanie układu.

Dane szeregowe są wprowadzane przez wejście *Szeregowe wejście danych* do komórek *BSC* o numerach 6, 5 i 4 podczas zwykłej pracy układu. Na linię sterującą *Wybór trybu testowego* jest podawany następnie stan wysoki. Powoduje to, że dane wprowadzone do komórek 6, 5 i 4 podawane są na wewnętrzne układy przez linie *Wyjście danych*. Następnie stan linii *Przesuń/Ładuj* zostanie zmieniony na niski, co spowoduje podanie sygnałów wyjściowych układów wewnętrznych na wejścia *D* przerzut-



Rys. 5.

ników komórek *BSC* 1, 2 i 3. Impuls zegarowy powoduje zapisanie tych sygnałów w przerzutnikach komórek. Z kolei na linii *Przesuń/Ładuj* ponownie pojawia się stan 1, a na linii *Wybór trybu testowego* stan 0. Trzy kolejne impulsy zegarowe powodują wyprowadzenie informacji zawartej w komórkach, a więc informacji pobranej z wyjść układów wewnętrznych.

Mechanizm ten dobrze ilustruje następujący przykład - układ scalony z rys. 1 zawiera trzy inwertery, których wejścia i wyjścia znajdują się odpowiednio po lewej i prawej stronie schematu. Jeśli przez wejście *TDI* wprowadzona zostanie sekwencja 101xxx (sekwencja wprowadzana jest poczynając od LSB, x - stany nieistotne) przedstawiona na **rys. 3**, to na wejściach dolnego i górnego inwertera pojawią się "1" logiczne, natomiast na wejściu środkowego - "0" logiczne.

Po wygenerowaniu przez układ *TAP* sekwencji testowania, w komórkach *BSC* 1, 2 i 3 znajdą się uzupełnienia stanów poprzednio wprowadzonych do komórek 4, 5 i 6. Jeśli układy wewnętrzne dokonały poprawnej operacji, po odpowiednich sygnałach sterujących z układu *TAP* sekwencja wyjściowa będzie miała postać xxx010 (bit LSB sekwencji wyprowadzany jest pierwszy, x - stany nieistotne). Jakakolwiek inna sekwencja oznacza, że układ nie działa prawidłowo. Zgodnie z normą IEEE 1149.1 taka funkcja testowa nosi nazwę INTEST.

Inny, przydatny rodzaj testu, który można przeprowadzić wykorzystując układy *BSC*, nosi nazwę EXTEST. Wykorzystywany jest do testowania zewnętrznych połączeń między układami wyposażonymi w *BSC*. Test ten polega na załadowaniu do komórek sekwencji i sprawdzeniu czy w połączeniach między układami nie ma zwarć ani rozwarć.

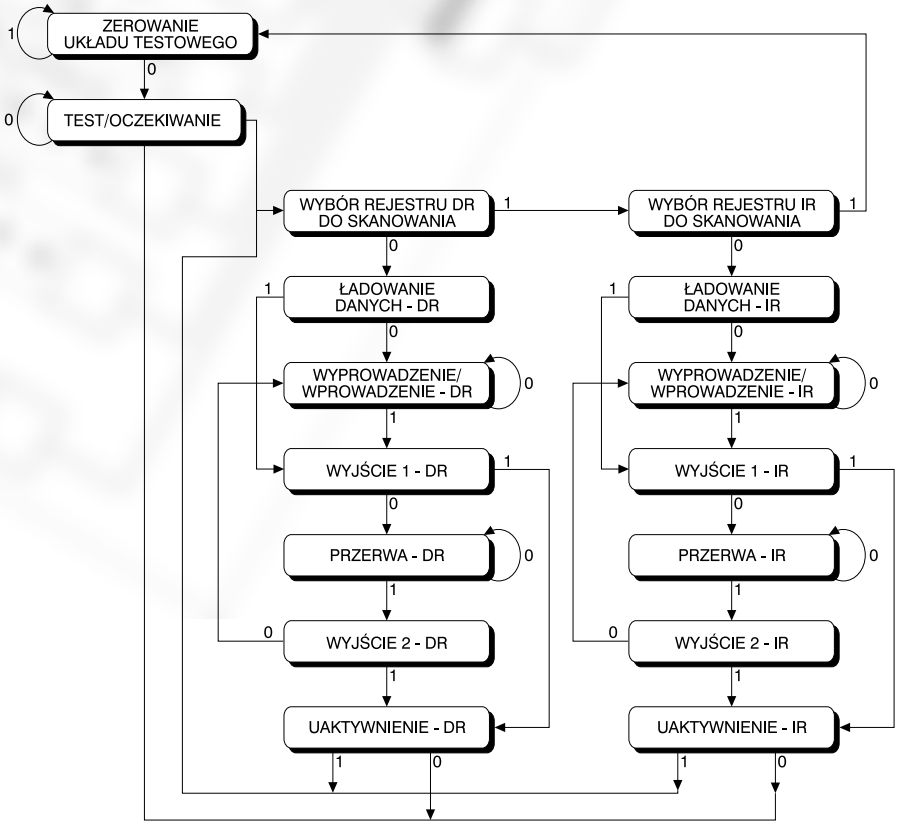
Na **rys. 4** przedstawiono dwa połączone ze sobą układy wyposażone w ścieżkę *BSC*. Dwa z połączeń zostały zwarte kroplą cyny, a trzecie jest zwarte do masy. Są to przypadki często spotykane w procesie produkcji i przy usuwaniu usterek układów. Oto jak system *BSC* może wykryć te błędy, wykorzystując funkcje SAMPLE/PRELOAD i EXTEST: do komórek *BSC* przy pomocy funkcji SAMPLE/PRELOAD ładowana jest sekwencja testowa o postaci xxxx 101x xxxx xxxx (x - stan nieistotny). Po uruchomieniu funkcji EXTEST na wyjściach komórek *BSC* 11 i 10 pojawiają się "1" logiczne, natomiast na wyjściu komórki *BSC* 11 - logiczne "0". Funkcja EXTEST zbiera następnie dane doprowadzone do wejść komórek 5, 6 i 7 drugiego układu. Dane te są następnie analizowane. Wyjściowa sekwencja testowa powinna mieć postać xxxx xxxx 1x01 xxxxx, ale z zebranych danych wynika, że jest to xxxx xxxx 0x11 xxxx. Logiczna "1", której obecność stwierdzono na wejściu komórki 6 jest nieprawidłowa, ponieważ wejście tej komórki zostało zwarte z wejściem komórki 5. Logiczne "0" na wejściu komórki 8 oznacza, że wejście to zostało zwarte z masą. Następnie na wejście komórek *BSC* jest podawana sekwencja testowa o postaci xxxx 010x xxxx xxxx. Wykonanie funkcji EXTEST pozwala stwierdzić, że sekwencja na wejściach drugiego układu ma postać xxxx xxxx 0x11 xxxx. Krótka analiza pozwala znaleźć rozstrzygnięcie: połączenie komórek 10 i 8 jest zwarte z masą, natomiast połączenia komórek 5, 6, 11 i 12 zostały zwarte ze sobą.

Organizacja układów *BSC* i kontroler *TAP*

Po przyjrzeniu się podstawom koncepcji testowania krawędziowego, kolejnym krokiem jest poznanie organizacji układów *BSC*. Schemat architektury układów *BSC*, zgodny z normą IEEE 1149.1, przedstawiono na **rys. 5**.

Zawiera ona trzy podstawowe bloki funkcjonalne:

- Kontroler *TAP*: jest to 16-stanowy automat, zrealizowany z użyciem mikrokontrolera, reagujący na sygnały podawane na *Test Access Port* i generujący podstawowe sygnały zegarowe i sterujące dla pozostałych bloków funkcjonalnych.
- *Rejestr instrukcji*: jest to szeregowy rejestr FIFO, do którego ładowany jest kod polecenia *BSC*. Kod ten wskazuje kontrolerowi *TAP*, jaki test należy przeprowadzić.
- *Rejestry danych testowych*: są to także szeregowe rejestry FIFO. Rejestr ten za-



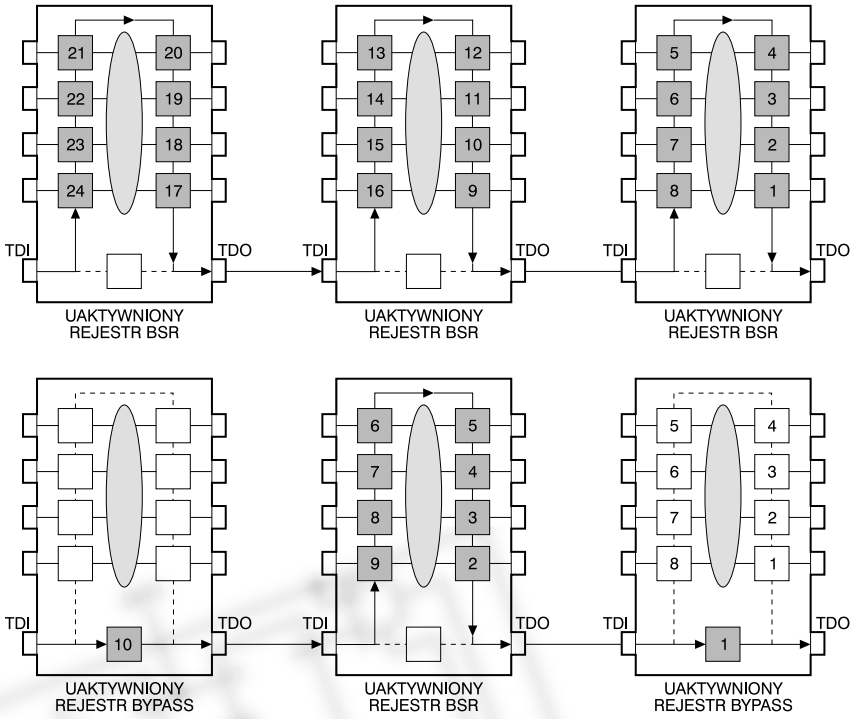
Rys. 6.

wiera wszystkie połączone łańcuchowo komórki *BSC*. Pozostałe rejestry z tej grupy to *Rejestr obejściowy* oraz *Rejestr identyfikacji układu*.

Trzy wyżej wymienione bloki współpracują ze światem zewnętrznym za pośrednictwem czterech (ewentualnie pięciu) linii we/wy. Są to: sygnał zegarowy (*TCK*), sygnał wyboru trybu pracy (*TMS*), testowe dane wejściowe (*TDI*), testowe dane wyjściowe (*TDO*) oraz - opcjonalnie - zerowanie testu (TRST).

Oto skrócony opis poszczególnych sygnałów:

- Sygnał zegarowy (***TCK***), taktujący kontroler *TAP*, jest całkowicie niezależny od wszystkich innych sygnałów zegarowych, które mogą być doprowadzane do układów wewnętrznych układu zgodnego z normą IEEE 1149.1. Zbocze narastające *TCK* inicjuje ładowanie informacji znajdujących się na wejściach *TMS* i *TDI*, natomiast zbocze opadające powoduje wyprowadzenie informacji na wyjście *TDO*. Inaczej mówiąc, dane są wprowadzane do komórek *BSC* zboczem narastającym sygnału *TCK*, wyprowadzane zaś zboczem opadającym tego sygnału.
- Sygnał selekcji trybu testowego (***TMS***) - na wejście to jest podawana sekwencja zer i jedynek, wprowadzana następnie do kontrolera *TAP*. Na podstawie sekwencji kontroler przyjmuje jeden z 16 stanów, i generuje odpowiadające temu stanowi sygnały taktujące i sterujące wszystkie pozostałe części układu *BSC*.
- Wejście danych testowych (***TDI***): jest to szeregowe wejście danych, którymi mogą być instrukcje lub informacje przeznaczona do załadowania do układów *BSC*. Wprowadzanie odbywa się poczynając od LSB. Liczba wprowadzanych bitów jest zależna od liczby komórek *BSC* oraz kodu wprowadzanej instrukcji. Dane są zatrzaskiwane w rejestrze zboczem narastającym *TCK*.
- Wyjście danych testowych (***TDO***): jest to szeregowe wyjście danych, na które kontroler *TAP* wyprowadza wyniki testowania lub instrukcję. Dane taktowane są zboczem opadającym sygnału *TCK*, a ich sekwencję rozpoczyna LSB. Jeśli nie jest dokonywana operacja wyprowadzania danych, wyjście to jest wprowadzane w stan wysokiej impedancji.
- Wejście zerowania testu (***TRST***, opcjonalne): norma IEEE 1149.1 stawia wymaganie, by układ z nią zgodny był inicjalizowany przez wprowadzenie w konkretny stan. Jest to stan *Test Logic Reset State* (stan wyzerowania logicznych układów testujących). Stan ten można wymusić podając na wejście *TCK* pięć impulsów zegarowych i utrzymując na wejściu *TMS* stan wysoki. Jednak norma przewiduje także możliwość wyzerowania układów niezależnie od stanu wejść *TCK* i *TMS*. Można to zrealizować dodając obwód zerowania układów testujących po włączeniu zasilania. Inna możliwość to uzupełnienie układu o wejście TRST.



Rys. 7.

Kontroler *TAP*

Kontroler *TAP* jest, jak już wcześniej wspomniano, 16-stanowym automatem skończonym (ma określone wszystkie możliwe stany), który działa zgodnie ze schematem przedstawionym na **rys. 6**. Stany, których nazwy zawierają znaki „-DR“ dotyczą operacji na rejestrach danych. Oznacza to, że kontroler dokonuje pewnej operacji, określonej przez zawartość rejestru instrukcji, na jednym z rejestrów danych. Stany, których nazwy zawierają znaki „-IR“ dotyczą operacji na rejestrze instrukcji.

Warunek logiczny podany obok nazwy stanu ("1" lub "0") wskazuje, jaką wartość musi mieć linia *TMS* w momencie wystąpienia następnego zbocza narastającego sygnału *TCK*, by doszło do przejścia do następnego stanu automatu. Cykl taktowania kontrolera *TAP* obejmuje czas od pojawienia się zbocza narastającego *TCK* do zbocza opadającego tego sygnału.

Diagram stanów zawiera sześć stanów stabilnych: *Test-Logic-Reset* (zerowanie testowych układów logicznych), *Run-Test/Idle* (Test/Oczekiwanie), *Shift-DR* (przesunięcie zawartości rejestru danych), *Pause-DR*, *Shift-IR* (przesunięcie zawartości rejestru instrukcji), *Pause-IR*. Należy zwrócić uwagę na to, że gdy na linii *TMS* panuje stan wysoki, możliwy jest tylko jeden stan stabilny - jest to stan *Test-Logic-Reset*. Oznacza to, że jeśli na linii *TMS* panuje stan wysoki, wyzerowanie układów ścieżki krawędziowej nastąpi po podaniu pięciu impulsów *TCK*.

Po włączeniu zasilania lub podczas normalnej pracy układu scalonego kontroler *TAP* jest wprowadzany w stan wyzerowania przez podanie 1 na linię *TMS* oraz pięciu impulsów na linię *TCK*. Następnie kontroler generuje sygnał, który wprowadza układy ścieżki krawędziowej w stan umożliwiający normalną pracę układu. Gdy powstaje potrzeba przeprowadzenia testu, na wejścia *TMS* i *TCK* jest podawana sekwencja powodująca przejście kontrolera *TAP* przez pożądane stany.

Stany bloków odnoszących się do rejestrów danych (*DR*) lub rejestru instrukcji (*IR*) są takie same. Pierwsza operacja po wejściu do dowolnego z tych bloków to załadowanie informacji. W stanie *Capture-DR* kontroler dokonuje załadowania danych do wybranej ścieżki danych. Jeśli wybranym rejestrem jest rejestr BSR, wprowadzane są do niego stany wejść danych układu. W stanie *Capture-IR* kontroler dokonuje wprowadzenia stanu układów ścieżki krawędziowej do rejestru instrukcji.

Ze stanu *Capture* kontroler *TAP* przechodzi do stanu *Shift* (Przesuwanie) lub *Exit1* (Wyjście 1). Na ogół stan *Shift* następuje po stanie *Capture* i dane testowe lub informacja o statusie mogą być wyprowadzone na zewnątrz celem analizy, a nowe dane wprowadzone do układu. Po przeprowadzeniu operacji właściwych stanowi *Shift*, kontroler przez stany *Exit1* i *Update* powraca do stanu *Run-Test/Idle* lub przez stan *Exit1* przechodzi do stanu *Pause*. W stanie *Pause* zatrzymano przesuwanie informacji przez rejestry danych lub instrukcji, w celu przeprowadzenia innej wymaganej operacji, np. ładowania pamięci buforowej testera. Przesuwanie może być następnie ponownie zainicjowane po przejściu ze stanu *Pause* do stanu *Shift* przez stan *Exit2* lub zaniechane przez przejście do stanu *Run-Test/Idle* przez stany *Exit2* i *Update*.

Rejestry wymagane przez standard

Norma IEEE 1149.1 narzuca obecność kilku rejestrów, a kilka innych proponuje jako opcjonalne: *Instruction Register* (Rejestr instrukcji), *Boundary Scan Register* (Rejestr ścieżki krawędziowej), *Bypass Register* (Re-

jestr obejścia) i *Device Indentification Register* (Rejestr identyfikacji układu).

Instruction Register (obowiązkowy) zawiera adresy i sygnały, sterujące niezbędne do włączenia wybranego rejestru danych w ścieżkę testową. Kontroler *TAP* dokonuje operacji na tym rejestrze znajdując się w dowolnym ze stanów IR.

Instruction Register zawiera rejestr przesuwny FIFO i rejestr instrukcji typu zatrzask. Jeśli kontroler otrzymuje sygnał Reset, ustawia w rejestrze instrukcji stany „1". Wymusza to na układach ścieżki krawędziowej normalny tryb pracy i włącza Bypass *Register* (lub *Device Indentification Register*) między wejście *TDI* i wyjście *TDO*. Układ zgodny z normą IEEE 1149.1 posiada dwa rejestry danych. Są to *Bypass Register* i *Boundary Scan Register*. Opcjonalny jest trzeci rejestr o nazwie *Device Indentification Register*. Rejestry te włączane są między wejście *TDI* a wyjście *TDO*.

Rejestr Instrukcji podaje adres umożliwiający dostęp do jednego z rejestrów danych, gdy kontroler *TAP* znajduje się w stanie skanowania rejestrów danych. Na podstawie sygnału sterującego z kontrolera *TAP* jest dokonywana selekcja wyjścia rejestru danych, które zostanie dołączone do wyjścia *TDO*. Selekcja jednej z linii rejestrów danych oznacza, że wszystkie inne linie pozostają w swych dotychczasowych stanach.

Rejestr ***Boundary Scan Register*** zawiera komórki zorganizowane w ścieżkę wokół wejść i wyjść funkcjonalnej części układu scalonego.

Bypass Register (obowiązkowy) zawiera tylko 1 bit. Po otrzymaniu sygnału zezwolenia rejestr ten tworzy jednobitowe połączenie między *TDI* i *TDO*. Rejestr ten pozwala na ominięcie ścieżki krawędziowej układów, które nie są objęte danym testem.

Załóżmy, że mamy do czynienia z układami scalonymi połączonymi jak na **rys. 7a.** Jeśli wszystkie komórki ścieżki krawędziowej zostaną uaktywnione, cała długość ścieżki wyniesie 24 bity. Jeśli jednak testowi powinien zostać poddany tylko układ znajdujący się w środku, należy tak skonfigurować ścieżkę testu, by pierwszy i ostatni układ scalony wprowadzały do tej ścieżki tylko jeden bit (rys. 7b). W efekcie ścieżka będzie zawierać tylko 10 bitów (8 w środkowym i po 2 w układach zewnętrznych), a nie 24. Czas testowania staje się dzięki temu o 58% krótszy. Problem nabiera innego wymiaru w przypadku procesora Pentium, którego ścieżka krawędziowa zawiera około 170 komórek. Rejestr *Bypass* jest wybierany, gdy w *Rejestrze instrukcji* znajdują się same jedynki.

Rejestr identyfikacji układu (opcjonalny) zawiera informację o producencie układu, numerze układu, jego wersji i inne dane dotyczące układu. Po zaadresowaniu zawartość tego rejestru może oczywiście być wyprowadzona na zewnątrz układu. Jest to bardzo przydatne dla stwierdzenia czy w danym gnieździe karty znajduje się właściwy układ.

Instrukcje standardu

Norma IEEE 1149.1 wymienia dziewięć instrukcji wykorzystywanych przez kontroler *TAP*, z których trzy winny być zaimplementowane w układzie, a pozostałych sześć jest opcjonalnych. Trzy pierwsze to instrukcje: *BYPASS*, *SAMPLE/PRELOAD* i *EXTEST*. Instrukcje opcjonalne to: *INTEST*, *RUN-BIST*, *CLAMP*, *HIGHZ*, *IDCODE* i *USERCODE*. Dokładniej przedstawione zostaną tylko instrukcje wymagane przez normę.

Instrukcja ***BYPASS*** pozwala układowi funkcjonować normalnie i włącza rejestr *BYPASS* między linie *TDI* i *TDO*. Dane testowe są przekazywane przez układ nie wpływając na jego działanie. Kod tej instrukcji składa się z samych jedynek.

Instrukcja ***SAMPLE/PRELOAD*** pozwala układowi funkcjonować normalnie i włącza *Rejestr ścieżki krawędziowej* między linie *TDI* i *TDO*. Instrukcja ta umożliwia zanalizowanie zawartości tego rejestru po wprowadzeniu kontrolera *TAP* w stan przeglądania danych. Instrukcja ta jest wykorzystywana także do załadowania danych testowych do rejestru Boundary Scan Register przed wykonaniem polecenia *EXTEST*. Kod instrukcji *SAMPLE/PRELOAD* określony jest przez producenta układu i podany w danych technicznych.

Instrukcja ***EXTEST*** wprowadza układ w tryb testu zewnętrznego i włącza *Rejestr ścieżki krawędziowej* między linie *TDI* i *TDO*. Dane zawarte w rejestrze są wyprowadzane na fizyczne wyprowadzenia układu scalonego, ewentualnie zewnętrzne dane testowe są wprowadzane do rejestru. Kod instrukcji *EXTEST* składa się z samych zer.

Tylko testowanie?

Po przeczytaniu tego artykułu można odnieść wrażenie, że JTAG jest wysokospecjalizowanym interfejsem służącym tylko do testowania układów cyfrowych. Pierwotnie rzeczywiście tak było.

W chwili obecnej JTAG jest wykorzystywany także do programowania układów programowanych w systemie ISP. Od dłuższego czasu dostępne są programowane w systemie programowalne struktury logiczne (FPGA i CPLD), a także coraz większa liczba procesorów i mikrokontrolerów, które mogą programować się same, bez konieczności stosowania dodatkowych, często drogich narzędzi.

Ze względu na rosnące znaczenie na rynku elektroniki nowoczesnych struktur logicznych ISP, coraz większego znaczenia będzie nabierało maksymalne ułatwienie ich programowania. Tak więc, dzięki interfejsowi JTAG nie trzeba będzie już wkrótce kupować specjalizowanego programatora (często z szeregiem przystawek). Co więcej - po zmontowaniu urządzenia składającego się z wielu układów ISP z wbudowanym interfejsem JTAG (ich typ i możliwości są bez znaczenia) można je zaprogramować jednocześnie, co znacznie skraca czas produkcji!

Piotr Zbysiński, AVT

Artykuł napisano w oparciu o materiały firm:

- *Altera (JTAG standard on FLEX devices),*
- *Intel (JTAG interface in Pentium testing),*
- *Texas Instruments (JTAG Support),*
- *Xilinx (CPLD Data Book 1997).*

Literatura dodatkowa:

1. „The Test Access Port & Boundary Scan Architecture" M. Maunder i Rodham E.Tullosa (IEEE Computer Society Press 1996).
2. „Standard Access Test Port and Boundary Scan Architecture", IEEE Std 1149.1 (1994).